Python
Lesson 2: Variables, Operations, and Types

Thomas Schwarz, SJ
Marquette University
Variables and Types

- All program languages specify how data in memory locations is modified

- Python: A variable is a handle to a storage location
  - The storage location can store data of many types
    - Integers
    - Floating point numbers
    - Booleans
    - Strings
Variables and Types

- Assignment operator \( = \) makes a variable name refer to a memory location.

- Variable names are not declared and can refer to any legitimate type.

```plaintext
a = 3.14156432
b = "a string"
```

- Create two variables and assign values to them.

- Variable \( a \) is of type floating point and variable \( b \) is of type string.

- After reassigning, both variable names refer to the same value.

- The floating point number is garbage collected.
Expressions

- Python builds expression from smaller components just as any other programming language
  - The type of operation expressed by the same symbol depends on the type of operands
- Python follows the usual rules of precedence
  - and uses parentheses in order to express or clarify orders of precedence.
Expressions

- Arithmetic Operations between integers / floating point numbers:
  - Negation (-), Addition (+), Subtraction (-), Multiplication (*), Division (/), Exponentiation (**)
  - Integer Division //
  - Remainder (modulo operator) (%)
Expressions

- IF we use / between two integers, then we always get a floating point number
- If we use // between two integers, then we always get an integer
  - a//b is the integer equal or just below a/b
Expressions

• Strings are marked by using the single or double quotation marks

• You can use the other quotation mark within the string

• Some symbols are given as a combination of a forward slash with another symbol
  
  • Examples: \t for tab, \n for new line, \’ for apostrophe, " for double quotation mark, \\ for backward slash
  
  • We’ll get to know many more, but this is not the topic of today
Expressions

• Strings can be concatenated with the +
• They can be replicated by using an integer and the * sign
• Examples:
  • "abc"+"def"  →  'abcdef'
  • 'abc\''+'fg'  →  'abc"fg'
  • 3*"Hi'"  →  "Hi'Hi'Hi'"
Change of Type

- Python allows you to convert the contents of a variable or expression to an expression with a different type but equivalent value
  - Be careful, type conversation does not always work
- To change to an integer, use `int()`
- To change to a floating point, use `float()`
- To change to a string, use `str()`
Example

• Input is done in Python by using the function `input`
  • Input has one variable, the prompt, which is a string
  • The result is a string, which might need to get processed by using a type conversion (aka `cast`)
  • The following prints out the double of the input (provided the user provided input is interpretable as an integer), first as a string and then as a number

```python
user_input = input("Please enter a number ")
print(2*user_input)
print(2*int(user_input))
```

```
Please enter a number 23
2323
46
```
• Python does not understand English (or Hindi) so giving it a number in other than symbolic form does not help

• It can easily understand “123”

• It does not complain about the expression having the same type.

```
>>> int("two")
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    int("two")
ValueError: invalid literal for int() with base 10: 'two'
>>> float("123")
123.0
>>> int(24)
24
```